PAT-NO: JP410040255A

DOCUMENT-IDENTIFIER: JP 10040255 A

TITLE: **HASH TABLE** CONTROL DEVICE

PUBN-DATE: February 13, 1998

INVENTOR-INFORMATION:
NAME
KURIBAYASHI, KENJI

ASSIGNEE-INFORMATION:
NAME COUNTRY
NEC SOFTWARE LTD N/A

APPL-NO: JP08198796

APPL-DATE: July 29, 1996

INT-CL (IPC): G06F017/30

ABSTRACT:

PROBLEM TO BE SOLVED: To avoid the deterioration of retrieving processing performance with respect to a **hash table**.

SOLUTION: With respect to the entry of the **hash table** 103, a key is controlled by binary tree structure and a register means 101 retrieves the **hash table** by a retrieving means 102. The retrieving means 102 retrieves the **hash table** and prepares a binary retrieving route recording means 105. At the time of registering unregistered key, the register means 101 balances the binary tree by a binary tree balancing means 104. The binary tree balancing means replace-processes a node by the binary retrieving route recording means 105 to balance the binary tree. Thereby a balanced binary tree is prepared without regard to a data registering **order**.

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平10－40255

(43)公開日　平成10年(1998)２月13日

(51)Int.Cl.[6]	識別記号	庁内整理番号	Ｆ Ｉ		技術表示箇所
Ｇ０６Ｆ　17/30			Ｇ０６Ｆ　15/411	３１０	

審査請求　有　　請求項の数３　ＯＬ　（全　７　頁）

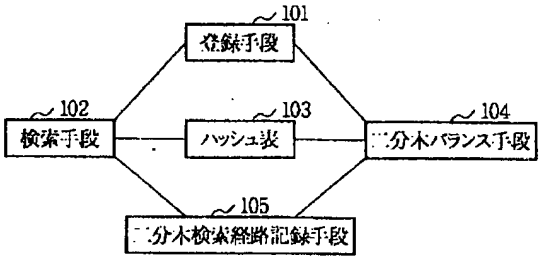
(21)出願番号　特願平8－198796

(22)出願日　平成８年(1996)７月29日

(71)出願人　000232092
日本電気ソフトウェア株式会社
東京都江東区新木場一丁目18番６号
(72)発明者　栗林　憲司
東京都江東区新木場一丁目18番６号　日本電気ソフトウェア株式会社内
(74)代理人　弁理士　京本　直樹　（外２名）

(54)【発明の名称】　ハッシュ表管理装置

(57)【要約】
【課題】　ハッシュ表に対する検索処理性能を低下させないようにする。
【解決手段】　ハッシュ表１０３のエントリに対し二分木構造でキーを管理し、登録手段１０１は、検索手段１０２によってハッシュ表を検索する。検索手段１０２はハッシュ表検索を行なうと同時に、二分木検索経路記録手段１０５を作成する。登録手段１０１は、未登録キーの登録時に、二分木バランス手段１０４によって二分木をバランスさせる。二分木バランス手段は、二分木検索経路記録手段１０５によって節の入れ換え処理を行ない二分木をバランスさせる。これにより、データの登録順序に関係なくバランスのとれた二分木を作成する。



1

【特許請求の範囲】

【請求項１】データベース管理システムにおけるハッシュ表管理装置において、

キーをハッシングした結果のハッシュ値によってキーを分類し、同一分類に含まれるキーを二分木構造で管理するハッシュ表と、

与えられたキーから前記ハッシュ表を検索する検索手段と、

二分木構造検索時にハッシュ表エントリから二分木構造をどのような順番で節を検索したかを記憶しておく二分木検索経路記録手段と、

検索の結果、該当するキーを管理する節が見つからなかったら新たにこのキーを管理する節を追加する登録手段と、

１つの節を追加した後に、前記二分木検索経路記録手段の記憶内容を逆順に参照しながら二分木をバランス処理する二分木バランス手段とを有することを特徴としたハッシュ表管理装置。

【請求項２】前記バランス処理は、当該節に対する前記二分木検索経路記録手段の記憶内容が左（右）なら、当該節の左（右）部分木の深さを、一つ前にバランス処理した節の左部分木の深さと右部分木の深さのうちの大きい方にプラス１した値とし、この値が２以上のときに行うことを特徴とする請求項１記載のハッシュ表管理装置。

【請求項３】前記バランス処理は、当該二分木構造のトップに至る全ての節について行うことを特徴とする請求項１記載のハッシュ表管理装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、データベース管理システムにおけるハッシュ表管理装置に関し、特にシノニム（同義語）の管理に関するものである。

【０００２】

【従来の技術】従来、この種のハッシュ表管理は、キーの衝突が多数発生しても検索効率を低下させないために、例えば特開平１－０９９１３１号公報に示されるように、同一のハッシュ値に変換された異なるキーを二分木構造でチェーンさせてハッシュ表に登録し、または検索する方式を採っている。

【０００３】

【発明が解決しようとする課題】しかし、上述した従来の技術では、二分木構造に新しいキーを登録するときに、常に木構造の末端の節の下に新しいキーを管理する節を作成するので、木構造にキーを登録する順序によって偏りができてしまうため、キーの登録順によって検索時の性能が低下するという問題点がある。特に、データベース管理システムにおいては、数十万あるいは数百万のオーダのレコードを扱うことと、キーの指定によっては上記のような偏りが発生しやすいという特徴があるので、上述の問題は、より深刻なものとなる。

2

【０００４】したがって、本発明の目的は、キーの登録順がハッシュ表検索の検索効率に影響を与えないようなハッシュ表管理装置を提供することにある。

【０００５】

【課題を解決するための手段】本発明の装置は、データベース管理システムにおけるハッシュ表管理装置において、キーをハッシングした結果のハッシュ値によってキーを分類し、同一分類に含まれるキーを二分木構造で管理するハッシュ表と、与えられたキーから前記ハッシュ表を検索する検索手段と、二分木構造検索時にハッシュ表エントリから二分木構造をどのような順番で節を検索したかを記憶しておく二分木検索経路記録手段と、検索の結果、該当するキーを管理する節が見つからなかったら新たにこのキーを管理する節を追加する登録手段と、１つの節を追加した後に、前記二分木検索経路記録手段の記憶内容を逆順に参照しながら二分木をバランス処理する二分木バランス手段とを有することを特徴とする。

【０００６】

【発明の実施の形態】次に、本発明の実施例について、図面を参照して説明する。

【０００７】図１を参照すると、本発明の実施の形態は、登録手段１０１と検索手段１０２とハッシュ表１０３と二分木バランス手段１０４と二分木検索経路記録手段１０５とを含む。

【０００８】ハッシュ表１０３は、各エントリに、キーを管理する二分木構造のルートのアドレスを持っている。図２には、ハッシュ表２０１の最初のエントリによって指定されるアドレスによって示されるルートの節２０２を起点に左右に分かれる二分木構造のルートが図示されている。二分木を構成する各節の構造は、図３に示すように、キー値と左部分木のポインタと、右部分木のポインタと、左部分木の深さと、右部分木の深さを情報に持つ。この節で管理するキー値より小さいキー値は、この節の左部分木の節で管理されている。また、この節で管理するキー値より大きいキー値は、この節の右部分木以下の節で管理されている。

【０００９】新たに節をチェーンする場合、登録手段１０１は、先ず与えられたキー値に対し、検索手段１０２を用いて検索（詳細は後述する）を行なう。検索の結果、未登録キーならば、このキーを管理する新たな節を作成し、検索手段１０２から返却された最後に比較した節のアドレスを参照し、その節の左部分木または右部分木に新たに作成した節をチェーンする。その後、二分木バランス手段１０４を用いて、節を追加したことによる各節における左右の部分木の深さの情報を更新すると共に、アンバランスを修正する。

【００１０】二分木バランス処理１０４は、二分木検索経路記録手段１０５を、記録順とは逆の順番に参照しながら、二分木検索経路記録手段１０５の指す節以下の部

分木に対し左右の部分木の深さを更新し、その結果、左右の部分木の深さの差が2以上になる節に対しては、深さの差が1以下になるようにバランス処理を行なう。

【0011】検索手段102は、与えられたキー値をハッシュし、ハッシュ値からハッシュ表103上のエントリを選択し、そのエントリに繋がっている二分木に対し、与えられたキー値による二分木検索を行なう。この二分木検索の結果、同一キーを管理する節が見つかれば、その旨のフラグとのぞ節のアドレスを返却し、見つからなければ最後に比較した節のアドレスを返却する。また、二分木検索時に辿った節のアドレスおよび左右どちらのポインタを参照したかを、二分木検索経路記録手段105に記録する。

【0012】次に、本実施例について、さらに詳細に説明する。

【0013】図2は、ハッシュ表エントリで管理するバランスされた二分木とその節の管理内容を示した説明図である。図2を参照すると、ハッシュ表の一つのエントリが二分木構造のルートの節202のアドレスを管理している。各節は、図3に示すように、左部分木のポインタ、左部分木の深さ、右部分木のポインタ、右部分木の深さ、およびキー値を情報として持つ。ルートの節202以下の二分木は、どの節を調べても、左右の部分木の深さの差は1以下であり、また、各節で管理するキー値よりも小さいキー値はこの節の左部分木以下の節で管理され、大きいキー値は右部分木以下の節で管理されている。

【0014】図4は、本発明を実施したシステムにおける、二分木検索経路記録手段105と、二分木に節を登録した後のバランス処理による二分木の変化を示した説明図である。図4を参照すると、二分木ポインタ301以下の二分木は、登録処理によって新たに登録する節303をチェーンした直後の状態である。二分木検索経路記録手段302は、二分木ポインタ301が指す節から順番に、節304のアドレスおよび右部分木検索の情報、節305のアドレスおよび右部分木検索の情報、節306のアドレスおよび左部分木検索の情報を記録している。この状態では、節303より上位にある節304の右部分木の深さ、節305の右部分木の深さ、節306の左部分木の深さは登録以前の状態である。

【0015】図5に示す二分木は、上記説明の二分木に対しバランス処理を施した結果の状態を表している。図4における節304およびその左部分木の状態は、節308およびその左部分木で表される通り変化しない。節303は節308の右部分木としてポイントされ、節309となる。節305は節309の左部分木としてポイントされ、節310となる。節306は節309の右部分木としてポイントされ、節311となる。

【0016】図6は二分木バランス手段104によるバランス処理の流れを示す図である。図6を参照すると、

401では、初期設定として、一つ前にバランス処理した節のポインタを保持するためのPRIORに、登録する節303のアドレスを設定する。

【0017】二分木検索経路記録手段105の配列を記録した順番とは逆に参照して、節（図4の例では306）のアドレスとこの節において左部分木を検索したか右部分木を検索したかの情報を取り出す（402）。このとき、全て参照し終ったら（403）、処理を終了する。二分木検索記録により左部分木検索か右部分木検索かをチェックし（404）、図4の例のように左部分木検索ならば、405にて当該節306の左部分木のポインタにポインタPRIORの保持するアドレスを設定する。また、当該節306の左部分木の深さについても、PROIRの指す節の左右の部分木の深さ（現時点では共に0）の大きい方の値＋1を設定する。逆に、右部分木検索であったなら、406において左部分木と同様に、右部分木のポインタと深さを設定する。以上の処理の結果、節306の左部分木の深さは1、右部分木の深さは0、左部分木のポインタは節303のアドレス、右部分木のポインタはNULLとなり、図4のように節306に節303がチェーンされたことになる。

【0018】上記処理の結果、407において現在の節306の左右の部分木の深さの差が2以上になったかどうかチェックする。図4の例では、深さの差は1である。深さの差が2以上ならば408でバランス処理ルーチンを呼び出しバランス処理する。バランス処理が不要なら現在の節306のアドレスをPRIORに設定し（409）、402の処理以降を繰り返す。

【0019】図4の例では、節305について以上の処理が繰り返される。今回は、404にて右検索と判断され、406にて、PRIORに保持された節306の左部分木の深さ1に＋1された2が現在の節305の右部分木の深さとなる。この結果、407にて、節305の左部分木の深さ0と、右部分木の深さの差が2以上となるのでバランス処理ルーチン408に入ることになる。

【0020】図7は、408で呼び出すバランス処理ルーチンの流れ図である。バランス処理は深さの差が2以上になる場合に動作するものなので、現在の節、その下位の節、さらにその下位の節の3レベルの節間でポインタチェーンを張り替える処理である。図7を参照すると、先ず、現在の節の検索経路を調べる（501）。

【0021】図4の例のように、現在の節305の検索経路が右ならば、さらに一つ前の節306の検索経路を調べる（505）。図4の例のように、一つ前の節308の検索経路が左ならば、506で、一つ前の節306の左部分木の節303をTOPの節とする。TOPの節306の左部分木には現在の節305を繋ぎ、右部分木には現在の節305の右部分木の節306を繋ぐ。TOPの節303に繋がっていた左部分木（図4の例ではNULL）は、TOPの節303の左部分木の節305の

5

右部分木して繋ぎ、またＴＯＰの節３０３に繋がっていた右部分木（図４の例ではＮＵＬＬ）は、ＴＯＰの節３０３の右部分木の節３０６の左部分木として繋ぐ。この結果、図４の二分木は図５のようにバランスする。

【００２２】５０５で一つ前の節の検索経路が右ならば、５０７で、現在の節の右部分木の節をＴＯＰにし、ＴＯＰの左部分木を現在の節の右部分木に繋ぎ、現在の節をＴＯＰの左部分木に繋ぐ。

【００２３】一方、５０１において、検索経路が左ならば、さらに一つ前の節の検索経路を調べる（５０２）。一つ前の節の検索経路が右ならば、５０３で、一つ前の節の右部分木の節をＴＯＰの節とする。ＴＯＰの右部分木には現在の節を繋ぎ、左部分木には現在の節の左部分木を繋ぐ。ＴＯＰに繋がっていた左部分木と右部分木は、それぞれ、ＴＯＰの左部分木の節の右部分木と、ＴＯＰの右部分木の節の左部分木として繋ぐ。

【００２４】５０２で一つ前の節の検索経路が左ならば、５０４で、現在の節の左部分木の節をＴＯＰにし、ＴＯＰの右部分木を現在の節の左部分木に繋ぎ、現在の節をＴＯＰの右部分木に繋ぐ。

【００２５】５０８で左右の部分木のポインタを繋ぎ替えた節について左右の部分木の深さについても更新する。最後に、ＴＯＰの節をバランス処理結果の節として呼び出し元に返却する（５０９）。

【００２６】なお、以上に説明した実施例においては、図６の４０７で、左部分木の深さと右部分木の深さが２以上の場合にバランス処理ルーチン４０８に入るが、これを３以上または４以上というように、処理ルーチンに入る機会を少なくしてもよい。

【００２７】また、図６の４０３で二分木検索経路記録

6

を逆順に参照するのを二分木構造のトップに至るまで行っているが、途中迄にしてもよい。

【００２８】

【発明の効果】本発明によれば、二分木構造をバランスすることにより、二分木の一部の部分木の深さが深くなり、その部分のツリーチェーンを辿る検索効率が悪くなることを防ぐことができる。

【図面の簡単な説明】

【図１】本発明のデータベース管理システムにおけるハッシュ表管理装置を示すブロック図である。

【図２】本発明のハッシュ表エントリで管理するバランスされた二分木例を示した説明図である。

【図３】本発明における節の構造を示す図である。

【図４】本発明における二分木検索経路記録手段と、二分木に節を登録した後のバランス処理前二分木を示した説明図である。

【図５】図４に示した二分木についてのバランス処理後の状態を示した説明図である。

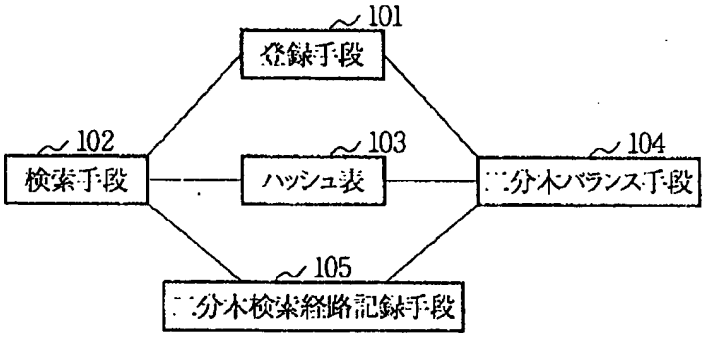
【図６】本発明を用いた二分木バランス処理の流れを示すフローチャートである。

【図７】図６の二分木バランス処理のなかで呼び出すバランス処理ルーチンの流れを示すフローチャートである。

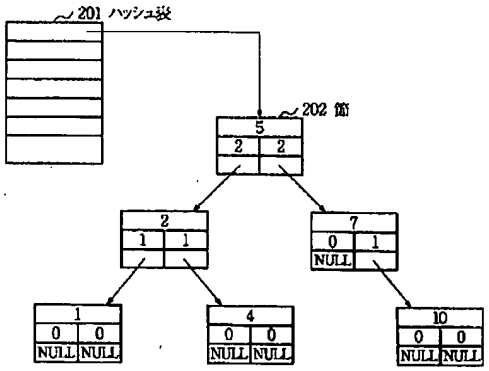
【符号の説明】

１０１　　登録手段
１０２　　検索手段
１０３，２０１　　ハッシュ表
１０４　　二分木バランス手段
１０５，３０２　　二分木検索経路記録手段

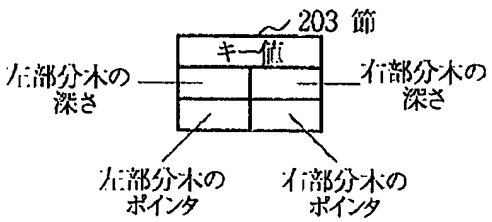
【図１】



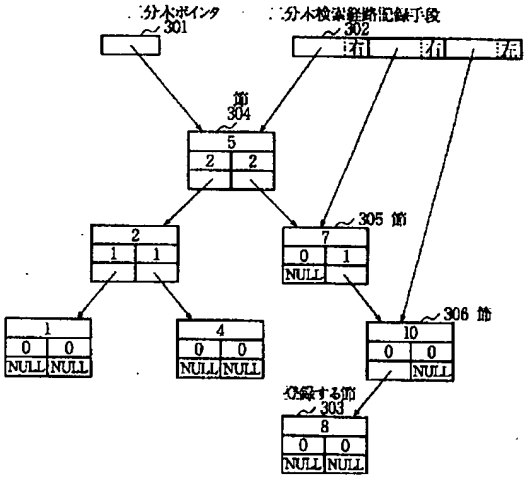
【図２】


201 ハッシュ表
202 節
5
2 2
2
1 1
7
0 1
NULL
1
0 0
NULL NULL
4
0 0
NULL NULL
10
0 0
NULL NULL

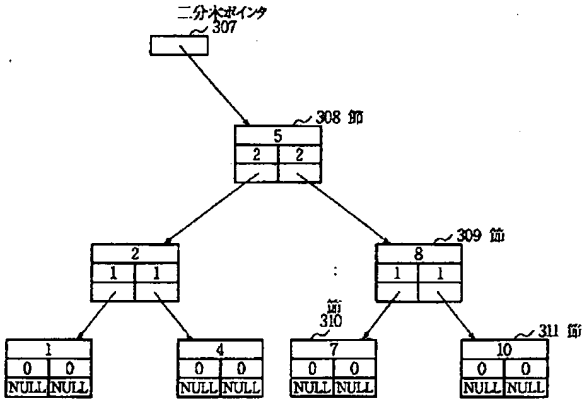
【図３】


203 節
キー値
左部分木の深さ
右部分木の深さ
左部分木のポインタ
右部分木のポインタ

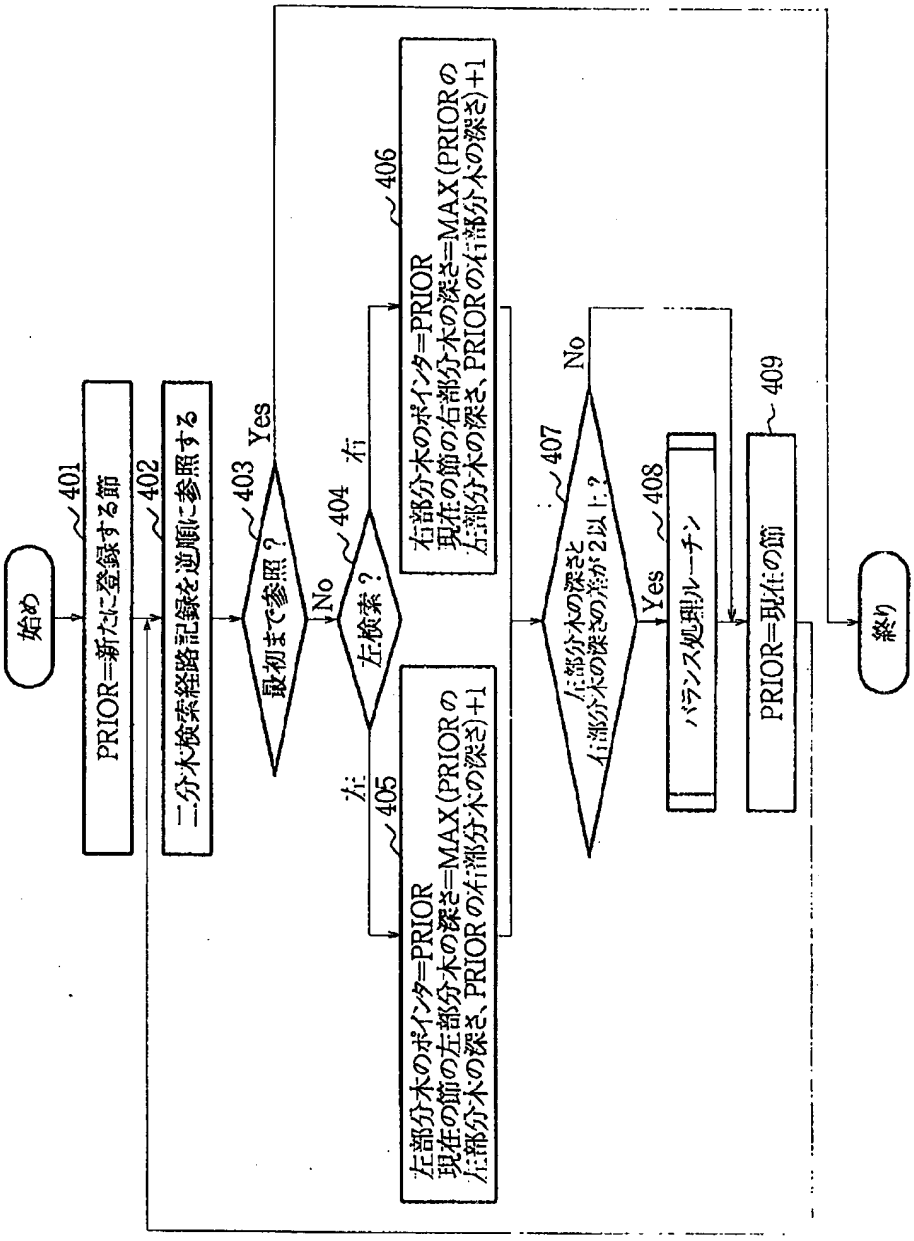
【図４】


二分木ポインタ
301
二分木検索経路記録手段
302
節
304
5
2 2
2
1 1
7
0 1
NULL
305 節
1
0 0
NULL NULL
4
0 0
NULL NULL
10
0 0
NULL
306 節
削除する節
303
8
0 0
NULL NULL

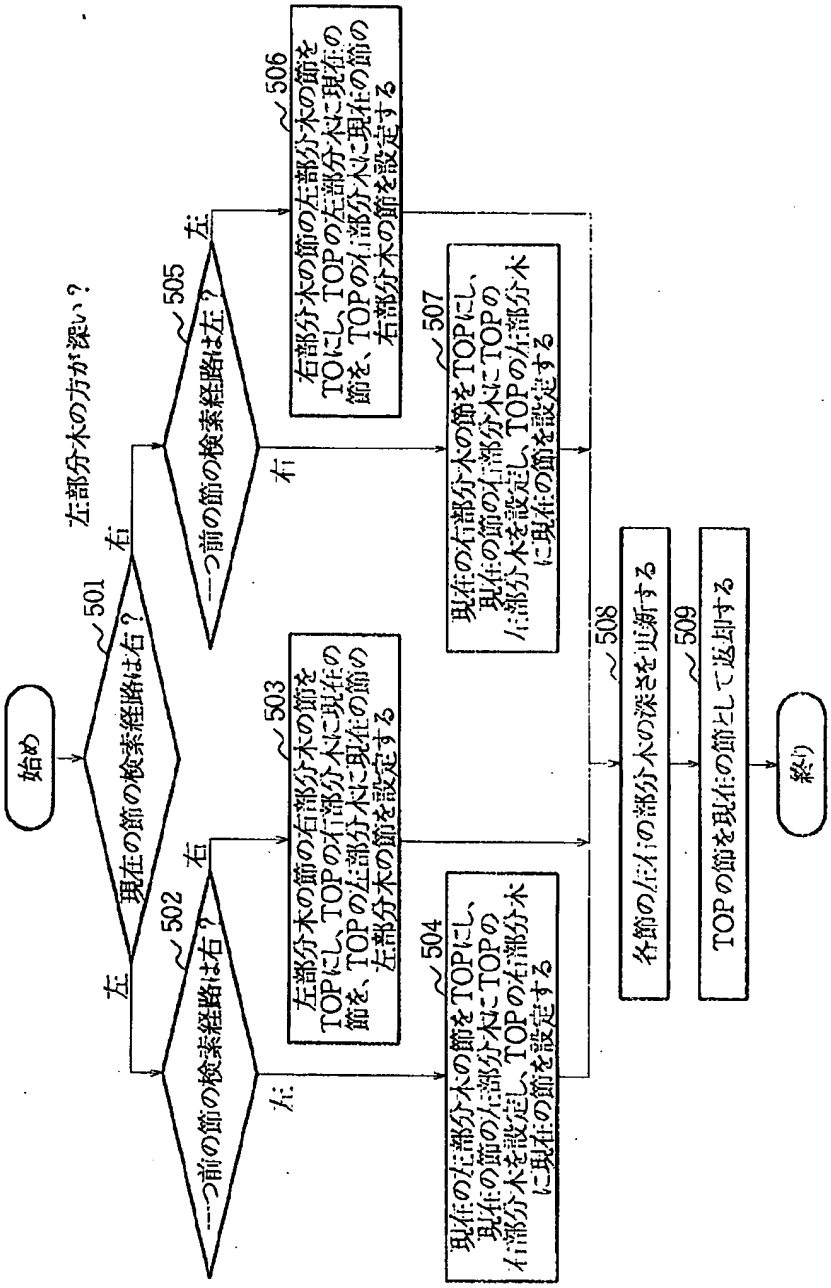
【図５】


二分木ポインタ
307
308 節
5
2 2
2
1 1
8
1 1
309 節
1
0 0
NULL NULL
4
0 0
NULL NULL
節
310
7
0 0
NULL NULL
10
0 0
NULL NULL
311 節

【図6】


始め
401
PRIOR=新たに登録する節
402
二分木検索経路記録を逆順に参照する
403
最初まで参照？
Yes
No
404
左検索？
左
右
405
左部分木のポインター=PRIOR
現在の節の左部分木の深さ=MAX(PRIORの
左部分木の深さ、PRIORの右部分木の深さ)+1
406
右部分木のポインター=PRIOR
現在の節の右部分木の深さ=MAX(PRIORの
左部分木の深さ、PRIORの右部分木の深さ)+1
407
左部分木の深さと
右部分木の深さの差が2以上？
No
Yes
408
バランス処理ルーチン
409
PRIOR=現在の節
終り

【図7】


始め
現在の節の検索経路は右？
501
左部分木の方が深い？
左
右
一つ前の節の検索経路は右？
502
右
左
左部分木の節の右部分木の節をTOPにし、TOPの左部分木に現在の節の節を、TOPの左部分木の節を設定する
503
現在の左部分木の節をTOPにし、現在の節の左部分木にTOPの右部分木を設定し、TOPの右部分木に現在の節を設定する
504
一つ前の節の検索経路は左？
505
左
右
右部分木の節の左部分木の節をTOPにし、TOPの左部分木に現在の節を、TOPの左部分木の節を設定する
506
現在の右部分木の節をTOPにし、現在の節の左部分木にTOPの左部分木を設定し、TOPの左部分木に現在の節を設定する
507
508
各節の左右の部分木の深さを更新する
509
TOPの節を現在の節として返却する
終り